Agentic AI Assignment 1

B. Tech 3rd Year

Semester: 6th

Session: 2026-2027

Submitted By:

Ayushi Tiwari (202334003)

Bhoomi Saxena (2023214249)

Bhavishya Bhardwaj (2023498559)

Submitted To **Mr. Ayush Kumar Singh**

Faculty Designation

Assistant Professor @ Sharda University

Noida, Uttar Pradesh, India

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING SHARDA

SCHOOL OF COMPUTING SCIENCE & ENGINEERING SHARDA

UNIVERSITY, GREATER NOIDA

Problem Statement:

Large organizations like TCS maintain detailed policy documents such as the Separation Policy in lengthy PDF formats. Employees often find it difficult and time-consuming to locate specific clauses related to resignation, termination, notice period, or redundancy.

Traditional search methods fail to understand contextual queries, and standalone language models may generate inaccurate responses without direct access to the official document.

Therefore, there is a need for an intelligent RAG-based system that can accurately retrieve relevant policy sections from the TCS Separation Policy document and generate precise, context-aware answers to user queries.

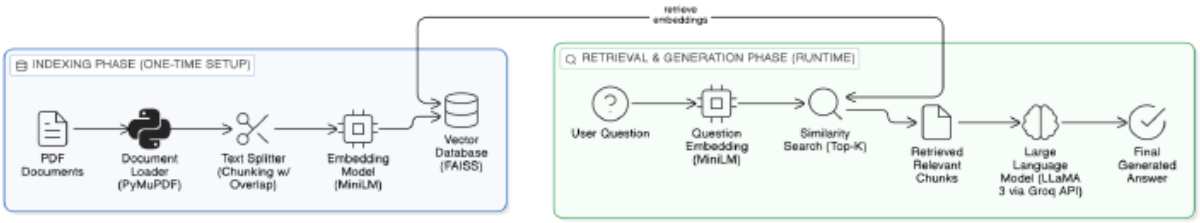
Dataset / Knowledge Source:

Type of Data:
The system uses unstructured textual data extracted from an official corporate policy document in PDF format. The primary knowledge source is the *TCS India Separation Policy (Version 2.0)* document, which contains detailed information about various types of employee separation such as resignation, termination, breach of contract, retirement, redundancy, and separation upon death.

The document includes structured sections, definitions, policy clauses, notice period rules, agreement commitments, and full & final settlement procedures. The textual content is extracted, divided into manageable chunks, converted into embeddings, and stored in a vector database to enable semantic retrieval.

Data Source:
The dataset used in this project is the *TCS India Policy – Separation* document. It is used strictly for academic and research purposes to demonstrate the implementation of a Retrieval-Augmented Generation (RAG) system for intelligent policy document querying and contextual answer generation.

RAG Architecture:



Text Chunking Strategy:

In this project, the TCS Separation Policy document is divided using a chunk size of approximately 800 characters with an overlap of 150 characters between consecutive chunks.

The policy document contains structured sections such as definitions, notice period rules, agreement commitments, and separation types. Therefore, the chunking process is designed to preserve logical flow while ensuring that each chunk contains meaningful and self-contained information.

The 150-character overlap ensures that important policy clauses, conditions, and references that span across chunk boundaries are not lost during splitting. This overlap maintains semantic continuity, especially in sections where legal or procedural conditions extend across multiple paragraphs.

Reason for Chosen Strategy:

Policy documents often contain interconnected clauses and conditional statements. Using slightly larger chunks (800 characters) helps retain complete rule explanations within a single chunk, improving contextual understanding during retrieval.

At the same time, controlled overlap ensures that boundary information is preserved without creating excessive redundancy. This balance enhances semantic similarity matching in the vector database and improves the accuracy of context-aware responses generated by the RAG system.

Embedding Details:

In this project, text embeddings are generated using the Sentence Transformers model: all-MiniLM-L6-v2. This pre-trained transformer model converts each text chunk from the TCS Separation Policy into a high-dimensional dense vector representation.

These vector embeddings capture the semantic meaning of the text rather than relying on simple keyword matching. As a result, the system can understand contextual relationships between user queries and policy clauses, even when the exact words do not match.

The generated embeddings are stored in a FAISS (Facebook AI Similarity Search) vector database, which enables efficient similarity comparison between user queries and stored document chunks. When a query is submitted, it is also converted into an embedding and matched against the stored vectors to retrieve the most relevant policy sections.

The use of semantic embeddings significantly improves the accuracy and contextual relevance of responses generated by the RAG system.

Reason for Selecting the Embedding Model:

The **all-MiniLM-L6-v2** model was selected due to its strong performance in semantic text understanding and information retrieval tasks. It generates context-aware embeddings that effectively capture the meaning of policy clauses and procedural statements within the TCS Separation Policy document.

The model is compact and computationally efficient, making it well-suited for academic implementations and systems running on limited hardware resources. Despite its lightweight architecture, it maintains reliable accuracy for semantic similarity search.

Additionally, the model integrates seamlessly with modern RAG pipelines using frameworks such as LangChain and vector databases like FAISS, ensuring smooth embedding generation and retrieval workflows.

By leveraging this model, the system achieves precise matching between user queries and relevant policy sections, significantly enhancing the contextual accuracy and reliability of generated responses.

Vector Database:

Vector Store Used: FAISS (Facebook AI Similarity Search)

In this project, FAISS is used as the vector database to efficiently store and retrieve semantic embeddings generated from the TCS Separation Policy document. FAISS is a high-performance similarity search library designed for fast nearest-neighbor search in large-scale vector datasets.

Once the document text is converted into dense embeddings using the Sentence Transformer model, these vectors are indexed within FAISS. This indexing enables rapid similarity comparison between stored document chunks and incoming user queries.

When a user submits a query, it is transformed into an embedding using the same embedding model. FAISS then performs a similarity search to identify the top *k* most relevant document chunks based on vector distance metrics. These retrieved chunks serve as contextual input to the Large Language Model, which generates an accurate and context-aware response.

The use of FAISS ensures scalable, efficient, and reliable retrieval performance within the Retrieval-Augmented Generation (RAG) pipeline.

Future Improvements:

Although the current RAG system effectively retrieves relevant information from the TCS Separation Policy document, several enhancements can further improve performance, scalability, and usability.

1. Intelligent Document Chunking

The present implementation uses fixed-size character-based chunking. In future iterations, advanced chunking strategies such as **semantic-aware splitting**, **section-based chunking**, or **recursive text splitting** can be implemented.

Since policy documents are naturally structured into headings and clauses, splitting content based on logical sections rather than character limits would preserve contextual integrity and improve retrieval precision.

2. Advanced Reranking Mechanism

Currently, document retrieval is based solely on vector similarity scores. Future improvements may include integrating a **cross-encoder reranking model** to refine the ranking of retrieved chunks.

This additional ranking layer would evaluate query–document relevance more precisely, improving answer quality. Furthermore, implementing **hybrid search** (combining semantic search with keyword-based search such as BM25) would balance contextual understanding with exact term matching, leading to higher retrieval accuracy.

3. Metadata-Enriched Retrieval

At present, all document chunks are treated uniformly. In future versions, structured metadata such as **section title, clause type, page number, separation category (Resignation, Termination, etc.)**, and document version can be stored alongside embeddings.

This would enable filtered retrieval and targeted search within specific policy sections, improving precision and enabling more controlled query responses.

4. Scalable Deployment and User Interface

The current implementation is notebook-based and designed for demonstration purposes. A production-ready version could include integration with a **web-based interface** using frameworks such as Streamlit, Flask, or React.

This would allow users to:

- Upload policy documents
- Ask natural language questions
- View retrieved policy sections
- Interact with the system in real time

Such deployment would enhance accessibility, usability, and real-world applicability of the RAG system.